5Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники института перспективной инженерии

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6
дисциплины «Программирование на Python»
«Работа с функциями в языке Python»
Вариант 12

Выполнила:
Коробка В.А.
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Воронкин Р.А.,
доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Цель: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Адрес репозитория: https://github.com/13r4lera/Python6.git

Был создан общедоступный репозиторий. Выбрана MIT License и язык программирования Python. Репозиторий был клонирован на локальный компьютер. В файл .gitignore добавлены правила PyCharm. Проработаны все примеры лабораторной работы. Все изменения были зафиксированы в репозитории.

Было выполнено индивидуальное задание 1 по варианту 12. Условие задания: Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены по алфавиту; вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

Была написана программа на Python, которая хранит данные о людях в списке словарей с ключами «Фамилия и имя», «Номер телефона» и «Дата рождения». Пользователь может добавлять записи, выводить весь список в виде таблицы и фильтровать людей по месяцу рождения. Список автоматически сортируется по алфавиту, а при отсутствии совпадений выводится соответствующее сообщение.

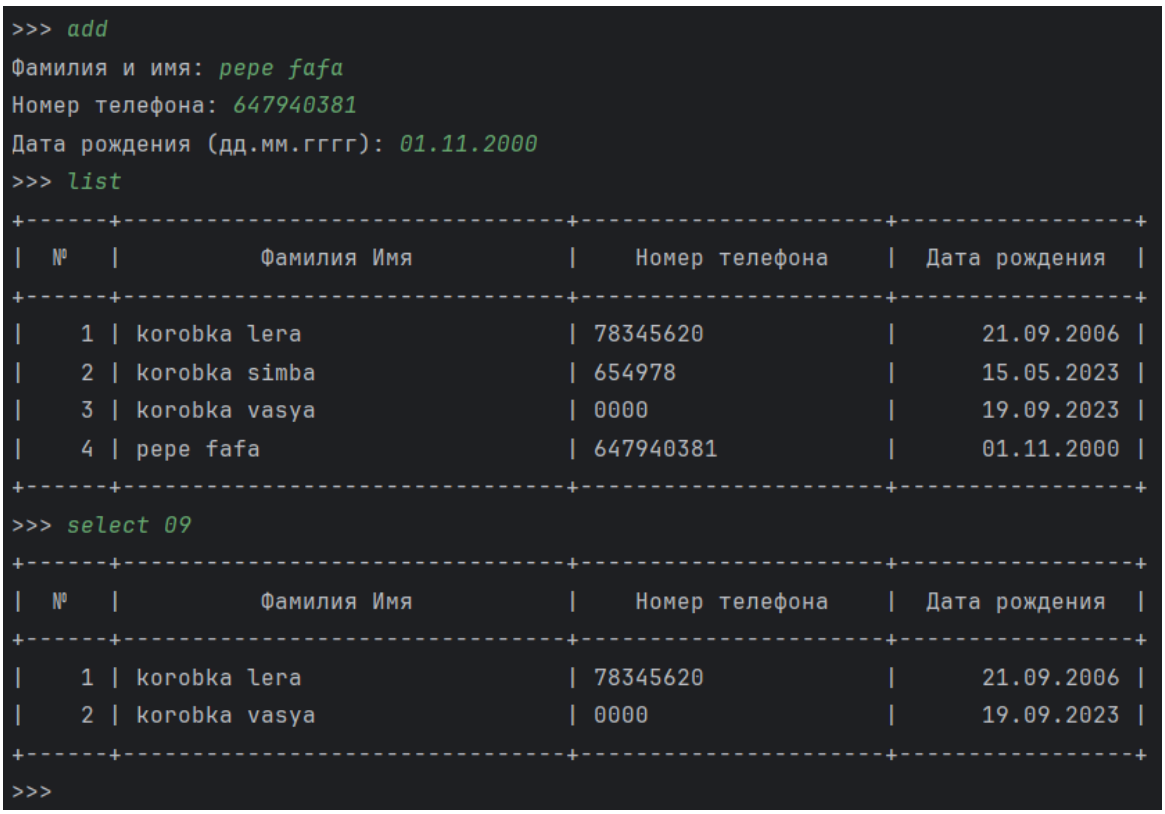

Рисунок 1. Результат выполнения первого задания

Полный код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
from datetime import date

def get_person():
    """
    Запросить данные о человеке
    """

    name = input("Фамилия и имя: ")
    phone = input("Номер телефона: ")
    day, month, year = map(int, input("Дата рождения (дд.мм.гггг): ").split('.'))
```

```
    birthday = date(year, month, day)

    return {
        "name": name,
        "phone": phone,
        "birthday": birthday
    }


def display_people(everybody):
    """
    Вывести список всех людей
    """
    if everybody:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "№",
                "Фамилия Имя",
                "Номер телефона",
                "Дата рождения"
            )
        )
        print(line)
```

```
            for idx, pers in enumerate(everybody, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                        idx,
                        pers.get('name', ''),
                        pers.get('phone', ''),

f"{pers['birthday'].day:02d}.{pers['birthday'].month:02d}.{pers['birthday'].year}"
                    )
                )

            print(line)

        else:
            print("Список людей пуст.")



    def birthday_select(persons, birthday_month):
        """
        Выбрать людей с днем рождения в определенном месяце
        """

        result = []
        for pers in persons:
            if pers['birthday'].month == birthday_month:
                result.append(pers)

        return result
```

```
def main():
    people = []

    while True:
        command = input(">>> ").lower()

        if command == "exit":
            break

        elif command == "add":
            person = get_person()
            people.append(person)
            if len(people) > 1:
                people.sort(key=lambda item: item.get('name', ''))

        elif command == "list":
            display_people(people)

        elif command.startswith('select'):
            parts = command.split(' ', maxsplit=1)
            birth_month = int(parts[1])

            selected = birthday_select(people, birth_month)
            display_people(selected)

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить человека;\n")
            print("list - вывести список людей;\n")
```

```
            print("select <месяц> - вывести имена людей, у которых день рождение в этом месяце;\n")
            print("help - отобразить справку;\n")
            print("exit - завершить работу с программой;\n")

        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


if __name__ == '__main__':
    sys.exit(main())
```

Было выполнено индивидуальное задание 2 по варианту 12. Условие задания: Реализуйте функцию format_all(template, *args, ** kwargs), которая применяет шаблон форматирования template. format () к каждому элементу из *args , подставляя значения из ** kwargs, и возвращает список строк.

Была создана программа, которая принимает строковый шаблон, набор позиционных аргументов и именованные аргументы через **kwargs. Для каждого позиционного аргумента функция подставляет его в шаблон, использует именованные значения из **kwargs, и собирает все полученные строки в список. В результате получается список строк, где шаблон применён ко всем позиционным аргументам.

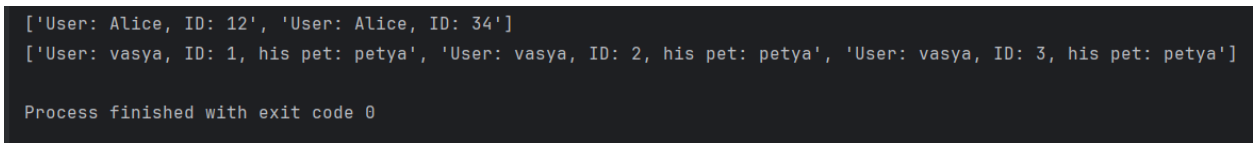

Рисунок 2. Результат выполнения задания 2

Полный код программы:

```
def format_all(template, *args, **kwargs):
    result = []

    for arg in args:
        res_str = template.format(arg, **kwargs)
```

```
        result.append(res_str)

    return result

if __name__ == '__main__':
    print(format_all("User: {name}, ID: {}", 12, 34, name="Alice"))
    print(format_all("User: {name}, ID: {}, his pet: {pet}", 1, 2, 3, name="vasya", pet="petya"))
```

Было выполнено индивидуальное задание 3 по варианту 12. Условие задания: Напишите функцию to_tuple(data), которая рекурсивно преобразует все списки внутри структуры в кортежи.

Была написана программа, которая рекурсивно проходит по структуре данных и преобразует все вложенные списки в кортежи. Каждый элемент списка проверяется: если это список, то вызывается рекурсивно to_tuple, если нет, то элемент добавляется без изменений. В результате получается структура с той же вложенностью, но все списки заменены на кортежи.

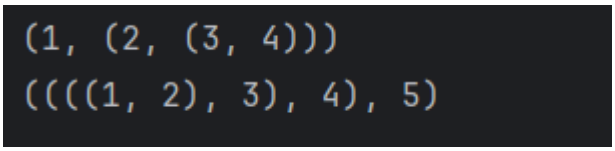

Рисунок 3. Результат выполнения задания 3

Полный код программы:

```
def to_tuple(data):
    result = []
    for item in data:
        if isinstance(item, list):
            result.append(to_tuple(item))
        else:
            result.append(item)

    return tuple(result)
```

```
if __name__ == "__main__":
    print(to_tuple([1, [2, [3, 4]]]))
    print(to_tuple([[[[1, 2], 3], 4], 5]))
```

Ответы на контрольные вопросы

1. Каково назначение функций в языке программирование Python?

Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.

2. Каково назначение операторов def и return?

Ключевое слово def сообщает интерпретатору, что перед ним определение функции. Если интерпретатор Питона, выполняя тело функции, встречает return, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

3. Каково назначение локальных и глобальных переменных при написании функций Python?

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды return.

5. Какие существуют способы передачи значений в функцию?

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции.

6. Как задать значение аргументов функции по умолчанию?

бывают параметры, которым уже присвоено значение по умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать. Тогда значение по умолчанию заменится на переданное.

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция.

8. Как осуществляется документирование кода согласно PEP257?

Все модули должны, как правило, иметь строки документации, и все функции и классы, экспортируемые модулем также должны иметь строки документации. Для согласованности, всегда используйте """triple double quotes""" для строк документации. Одиночные строки документации предназначены для действительно очевидных случаев. Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Вставляйте пустую строку до и после всех строк документации (однострочных или многострочных), которые документируют класс. Строки документации скрипта (самостоятельной программы) должны быть доступны в качестве "сообщения по использованию", напечатанной, когда программа вызывается с некорректными или отсутствующими аргументами

9. В чем особенность однострочных и многострочных форм строк документации?

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны умещаться на одной строке. Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием.

10. Какие аргументы называются позиционными в Python?

В простейшей функции мы просто сопоставляем позиции аргументов и параметров. Аргумент №1 соответствует параметру №1, аргумент №2 - параметру №2 и так далее.

11. Какие аргументы называются именованными в Python?

Если при объявлении функции назначить параметру значение по умолчанию - указывать соответствующий аргумент при вызове функции уже необязательно. Параметр становится опциональным. Опциональные параметры, кроме того, можно задавать при вызове функции, используя их имена.

12. Для чего используется оператор *?

Оператор «звёздочка» в Python способен «вытаскивать» из объектов составляющие их элементы.

13. Каково назначение конструкций *args и **kwargs?

Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины.

14. Для чего нужна рекурсия?

Рекурсия используется, когда задача естественно разбивается на одинаковые подзадачи, которые решаются аналогичным образом, например, вычисление факториала.

15. Что называется базой рекурсии?

База рекурсии — это условие, при котором функция не вызывает сама себя, а возвращает результат, останавливая рекурсивные вызовы.

16. Самостоятельно изучите, что является стеком программы. Как используется стек программы при вызове функций?

Стек программы хранит активные вызовы функций. Каждый вызов помещается в стек, а после завершения функции кадр удаляется, обеспечивая порядок LIFO.

17. Как получить текущее значение максимальной глубины рекурсии в языке Python?

Используется sys.getrecursionlimit(), который возвращает число рекурсивных вызовов до возникновения ошибки.

18. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python?

Появится ошибка RecursionError: maximum recursion depth exceeded, и выполнение программы прервется.

19. Как изменить максимальную глубину рекурсии в языке Python?

Используется sys.setrecursionlimit(значение), где значение - новая максимальная глубина рекурсии.

20. Каково назначение декоратора lru_cache?

Уменьшение количества лишних вычислений.

21. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?

Хвостовой вызов - это просто вызов рекурсивной функции, который является последней операцией и должна быть выполнена перед возвратом значения. Оптимизация хвостового вызова (TCO) - это способ автоматического сокращения рекурсии в рекурсивных функциях. Устранение хвостового вызова (TCE) - это сокращение хвостового вызова до выражения, которое может быть оценено без рекурсии. TCE - это тип TCO.

Вывод: были изучены функции в Python, рекурсия, позиционные и именованные аргументы, конструкции *args и **kwargs.